RAPPORT DE TIPE

PREAMBULE

Dans le but de chercher à optimiser le processus de reconstitution d'objets en trois dimensions, nous nous étions proposé de choisir et d'implémenter en Python un tel algorithme de reconstitution, et d'en étudier l'efficacité. C'est ce que nous avons fait, en choisissant l'algorithme dit d'« enveloppe visuelle » étudié notamment par B. Baumgart [1] et A. Laurentini [2]. Il présente l'avantage d'être conceptuellement assez simple, et d'être implémentable de manière relativement légère.

(72 mots)

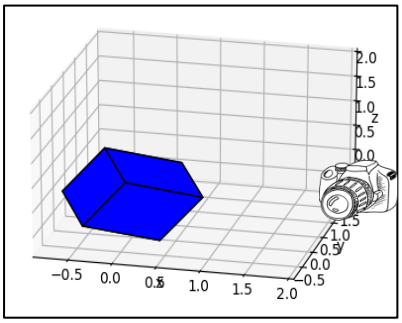
INTRODUCTION

Nous avons donc réalisé une implémentation de cet algorithme, puis avons effectué des tests pour en étudier l'efficacité, et pouvoir analyser la pertinence des choix réalisés durant la construction de l'algorithme. Je me suis personnellement occupé de la partie de modélisation en 3D du projet, tandis que mon binôme s'est occupé du prétraitement des photographies pour les rendre exploitables.

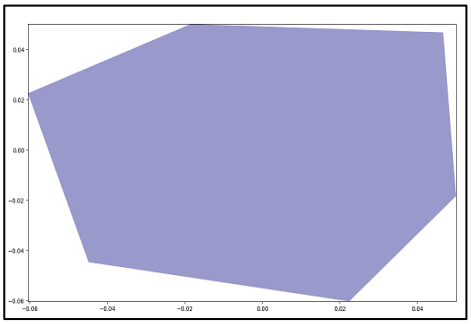
(72 mots)

CORPS PRINCIPAL

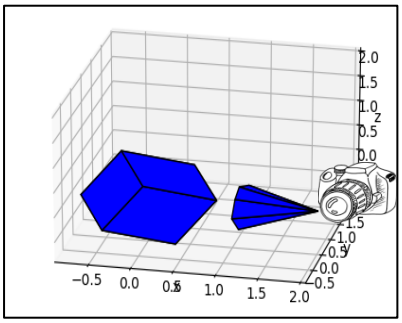
L' « enveloppe visuelle » d'un objet A est l'objet maximal qui donne la même silhouette que A quand on l'observe depuis un ensemble de points de vues donné. Ainsi, l'enveloppe visuelle de A depuis un point de vue unique est un cône dont le sommet est placé sur ce point de vue, et qui a la forme de la silhouette de A depuis ce point de vue (fig. 1).



(a)



(b)



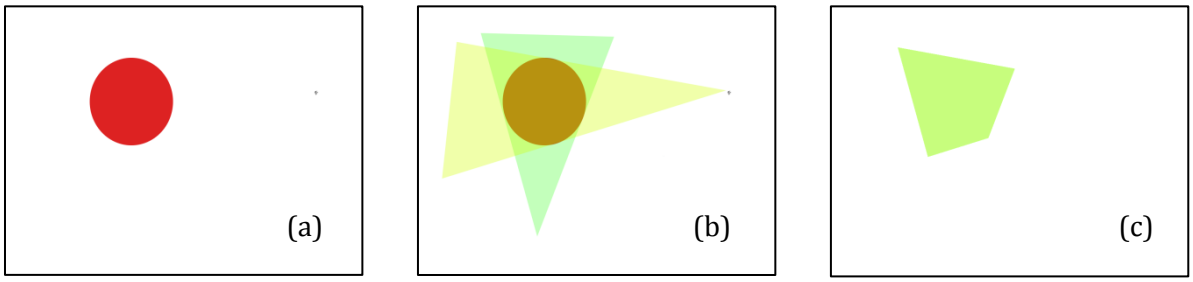
(c)

Fig. 1 : En (a), un cube et un point de vue (schématisé par l'appareil photo).

En (b), sa silhouette vue depuis le point de vue.

En (c), le cube et une partie du cône qui constitue son enveloppe visuelle pour le point de vue donné.

Pour obtenir l'enveloppe visuelle d'un objet depuis plusieurs points de vue, on peut alors simplement considérer l'intersection des enveloppes visuelles obtenues depuis chacun des points de vue. Un exemple (en deux dimensions, pour simplifier) est donné en fig. 2.



En (a), un objet.

En (b), les cônes correspondant aux deux points de vue.

En (c), l'intersection des deux cônes, soit l'enveloppe visuelle correspondant aux deux points de vue.

En multipliant les points de vue, l'enveloppe visuelle se rapproche de l'objet original. On peut donc utiliser cette méthode pour reconstituer le modèle 3D d'un objet, simplement à partir de photographies. Se dégagent deux étapes majeures pour cette reconstitution :

- Le traitement de la photographie, qui a pour but de dégager la silhouette de l'objet.
- La reconstitution à proprement parler, qui consiste à créer les cônes correspondant aux silhouettes, puis à en calculer l'intersection.

Ma contribution personnelle fut la construction d'un système de représentation en machine d'objets en 3D, ainsi que de tous les algorithmes l'accompagnant, en particulier les algorithmes d'intersection, de constructions de cônes, et de simulation de la prise d'une photographie. Cette construction fut réalisée en langage python, et des morceaux du code sont fournis en annexe.

La modélisation choisie s'appuie sur une approximation des objets par des polyèdres. Celle-ci présente l'avantage d'être assez légère en mémoire, car seuls quelques points importants y sont stockés.
Nous ne rentrerons pas les détails de la modélisation choisie et des algorithmes qui ont étés programmés, comme la fonction d'intersection de deux polyèdres, car les explications seraient assez longues et fastidieuses. Des extraits du code pourront cependant être retrouvés en annexe.

La modélisation construite nous a permis de réaliser des expériences. Ainsi j'ai réalisé des tests sur l'algorithme d'intersection pour obtenir des informations sur le temps d'exécution de cet algorithme pour divers complexités de polyèdres initiaux. Nous avons préféré une analyse du temps d'exécution pratique plutôt qu'une étude de la complexité théorique. En effet, si cette dernière donne des bonnes informations asymptotiques, elle ne donne aucun contrôle sur

la vitesse de convergence, et un algorithme en $O(n^2)$ pourra être en pratique beaucoup plus rapide à exécuter qu'un autre qui est en $O(n)$, simplement parce que les n considérés ne deviennent jamais très grands. De plus, une information sur le temps d'exécution, en secondes, est bien plus facilement interprétable qu'une information de complexité. Ainsi, des tests simples sur le programme que j'ai réalisé montre qu'une intersection simple, avec deux polyèdre ayant de l'ordre de la dizaine de sommets, prend en moyenne 0,15s sur un ordinateur portable, le processeur y étant cadencé à 2 GHz. Ceci, bien sûr, n'est que le temps d'exécution sur une machine particulière, et en utilisant des machines plus puissantes, ce temps pourrait être considérablement réduit. Mais il faut reprendre les objectifs que nous nous étions fixés : nous cherchions un algorithme efficace, qui renvoie des résultats les plus fidèles possibles tout ayant un temps d'exécution le plus court possible. Ici, les résultats obtenus signifient que pour pouvoir reconstituer un modèle 3D avec une précision acceptable, en un temps raisonnable (c'est-à-dire réaliser de l'ordre de 10 à 100 intersections de polyèdres qui possèdent de l'ordre de 1000 sommets), il faudrait une machine très puissante.

CONCLUSION GENERALE

En conclusion, le modèle construit ici est modérément efficace. Le traitement d'image effectué par mon binôme est, lui aussi, relativement lent (de l'ordre de la minute sur des ordinateurs personnels). Nous obtenons donc un programme qui fonctionne en un temps raisonnable, mais qui n'est pas aussi efficace que ce qu'on avait espéré.

BIBLIOGRAPHIE

[1] Bruce Guenther Baumgart : Geometric modeling for computer vision :
http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA002261
[2] Aldo Laurentini : The visual hull concept for silhouette-based understanding
http://areeweb.polito.it/ricerca/cgvg/Articles/pami94.pdf

ANNEXES